

Get content items

September 13, 2021 • Jan Cerman • 5 min read • TypeScript

While your copywriters draft articles and add finishing touches to the content in your project, you can deliver that content to web and mobile applications via an API. Learn how to retrieve your content from Kentico Kontent using the Delivery API.



The Delivery API is a read-only REST API that can serve your content in two modes: public and preview. In public mode, you can only retrieve content that is published and publicly available. In preview mode, you can retrieve published as well as any unpublished content.

This tutorial covers using the Delivery API in public mode, in which you don't need to authenticate your requests. Let's dive in and see what you need to retrieve a list of specific content items, such as articles.

Getting content items

To retrieve content items from a project, you will need the **project ID**. Your project is the primary organizational unit in Kentico Kontent. Every project is uniquely identified by an ID. You need to use the ID to tell the Delivery API where to look for content. For example, a project ID might look like this: `8d20758c-d74c-4f59-ae04-ee928c0816b`.

To get the ID of your project:

1. In Kentico Kontent, choose a project.
2. From the app menu, choose  **Project settings**.
3. Under **Environment settings**, choose **API keys**.
4. In the **Delivery API** box, click .

With the project ID, you can now make queries to the Delivery API using a URL such as `https://deliver.kontent.ai/<YOUR_PROJECT_ID>/items`.

TypeScript

```
1 // Tip: Find more about JS/TS SDKs at https://docs.kontent.ai/javascript
2 import { ContentItem, DeliveryClient } from '@kentico/kontent-delivery';
3
4 const deliveryClient = new DeliveryClient({
5   projectId: '8d20758c-d74c-4f59-ae04-ee928c0816b7'
6 });
7
8 deliveryClient.items<ContentItem>()
9   .toObservable()
10  .subscribe(response => console.log(response));
```

Calling the `/items` endpoint gives you all content items from the specified project in the form of a [paged list response](#) returned as JSON. Note that recently published items may appear in the Delivery API after a slight delay.

✓ Paging the results

If you don't need all content items at once, you can tinker with the paging by specifying the `limit` and `skip` query parameters.

For example, calling the `/items` endpoint with the `limit=3&skip=6` query parameters sets the page size to 3 and gives you the third page.

Filtering content items

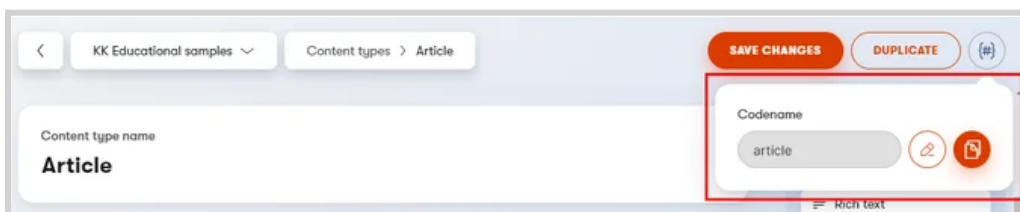
Now that you can get all content items from your project, you need to filter them to get only a specific few. In this example, you'll retrieve articles. These are the content items based on the *Article* content type.

To move any further, you need to find the **codename** of the *Article* content type.

i Quick facts about codenames

Codenames are identifiers of objects in Kentico Kontent. A codename is initially generated by the system from the object's name when it's saved for the first time, unless you specify one while adding the type.

You can copy codenames by clicking **(#)** near the name of a content type, content element, or other objects in your project. For content items, click **More actions > (#) Codename**.



Example: Displaying the codename of the *Article* content type.

Once you have the codename (in this case `article`) you can use it to filter the requested content items by their type.

The information about a content item's type is stored in the *System* object within its `type` property.

The *System* object contains metadata about the content item such as last content modification date, content type the item is based on, language, etc.

JSON

```
1  "system": {  
2    "id": "31f8470f-8a94-438a-8a47-f4cdb9c90ada",  
3    "name": "Why structured writing needs structured content",  
4    "codename": "structured_writing",  
5    "language": "en-US",  
6    "type": "article",  
7    "sitemap_locations": [],  
8    "last_modified": "2020-01-27T13:43:47.134249Z"  
9  }
```

To filter the content items by type, you need to compare the value in the `type` system property to `article` using the following notation: `system.type=article`. Any content items that are not based on the Article content type will be omitted from the response.

TypeScript

```

1 // Tip: Find more about JS/TS SDKs at https://docs.kontent.ai/javascript
2 import { DeliveryClient, TypeResolver } from '@kentico/kontent-delivery';
3 import { Article } from './models/Article';
4
5 const deliveryClient = new DeliveryClient({
6   projectId: '8d20758c-d74c-4f59-ae04-ee928c0816b7',
7   typeResolvers: [
8     // Create strongly typed models according to https://docs.kontent.ai/strongly-typed-models
9     new TypeResolver('article', (rawData) => new Article)
10   ]
11 });
12
13 deliveryClient.items<Article>()
14   .type('article')
15   .toObservable()
16   .subscribe(response => console.log(response));

```

The value comparison is done using the equals operator (`=`). You can also filter your content using many other criteria like an element value, publish date, and more. Check out [content filtering examples](#) to see what's possible.

Ordering content items

The Delivery API sorts content items alphabetically by their codenames by default. But with content like articles, you usually want to retrieve and display them in a certain order and get, for example, only three latest articles from your project.

When getting lists of content items, you can specify their order by using the `order` query parameter. The value of the `order` query parameter must be in the following format: `<PropertyToOrderBy>[<asc|desc>]`. Where the `PropertyToOrderBy` value specifies either a System property (such as `system.type`) or a content element within a content item (such as `elements.title`). For instance, if you don't specify the order when retrieving content, it is the equivalent of adding `order=system.codename[asc]` to your query.

To get three latest articles from your project, you need to provide the following query parameters:

- `system.type=article` – specifies the content type of the content items.
- `limit=3` – sets the number of content items to return (sometimes also referred to as page size).
- `order=system.last_modified[desc]` – sorts the content items by last modification date in descending order.

TypeScript

```

1 // Tip: Find more about JS/TS SDKs at https://docs.kontent.ai/javascript
2 import { DeliveryClient, SortOrder, TypeResolver } from '@kentico/kontent-delivery';
3 import { Article } from './models/Article';

```

```

3
4  const deliveryClient = new DeliveryClient({
5      projectId: '8d20758c-d74c-4f59-ae04-ee928c0816b7',
6      typeResolvers: [
7          // Create strongly typed models according to https://docs.kontent.ai/strongly-typed-
8      models
9          new TypeResolver('article', (rawData) => new Article)
10     ]
11 });
12
13 deliveryClient.items<Article>()
14     .type('article')
15     .limitParameter(3)
16     .orderParameter('system.last_modified', SortOrder.desc)
17     .toObservable()
18     .subscribe(response => console.log(response));

```

What's next?

You've learned how to get specific content from your Kentico Kontent project with filtering and sorting. Besides fetching content items, you can also use the [Delivery API](#) to get content types, elements, and taxonomies.

- Future-proof your app with [best practices on getting content](#).
- [Set up content preview](#) so that editors can preview unpublished content.
- Map your project's content types to [strongly typed models](#) to streamline your development process.
- [Share content between projects](#) either directly in the UI or programmatically by leveraging Kontent Delivery SDKs.
- See the [Kontent status page](#) to check the availability of Kontent APIs and the app in case you get an unexpected error.

Want to create an SDK for your preferred technology? Check out our [guidelines for SDK developers](#).